Introduction to Nuclear Programming

21F Midterm 2

Jeongbin Kim

June 6, 2024

Midterm 2

Question 1. (30 points) Write code `main01.f90`, which works as below:

- Construct subrotine `XOR(i1,i2,i3)` and `i1, i2` and `i3` are 8-byte integers.
- `i1, i2` are inputs. Set the bit of `i3` to 0 if the corresponding bit of `i1` and `i2` are the same and 1 if they are different.
  For example,

i1	0	1	0	0	1	1	0	1
i2	1	1	0	1	0	1	0	0
i3	1	0	0	1	1	0	0	1

- Check that when `i1`=77 and `i2`=212, `i3` is evaluated as 153.
- Leave the check process in the code.

My answer:

```
! 2024.2.24. Jeongbin kim
! N.E. Engineering Programming midterm2 (Fall 2021)
! Problem 1, ver2

module bin_dec_conv_module
    implicit none
contains
! deciaml to binary
    subroutine dec_to_bin(dec, bin)
        implicit none
        integer, parameter :: i8 = selected_int_kind(8) ! <- you can put this at the
    starting
        integer(i8), intent(in) :: dec                  !    of the module (Global Variable)
        integer, intent(out), dimension(1:8) :: bin
        integer(i8) :: local_dec
   ! maybe i can put intent(inout)
   ! in dec to not set a local_dec??  <- nope, this causes error:
        integer :: i

        local_dec = dec
        do i = 1, 8
            if (mod(local_dec,2) == 0 ) then
                bin(9-i) = 0
            else
                bin(9-i) = 1
            endif
            local_dec = local_dec/2
        enddo
    end subroutine dec_to_bin

! binary to decimal
    subroutine bin_to_dec(bin, dec)
        implicit none
        integer, parameter :: i8 = selected_int_kind(8)
        integer, intent(in), dimension(1:8) :: bin
```

```
35          integer, intent(out) :: dec
36          integer :: i
37
38          do i = 1, 8
39              dec = dec + bin(i) * ( 2**(8-i) )
40          enddo
41      end subroutine bin_to_dec
42  end module bin_dec_conv_module
43
44  module xor_module
45      use bin_dec_conv_module
46      implicit none
47      integer, parameter :: i8 = selected_int_kind(8)
48  contains
49      subroutine XOR(i1, i2, i3)
50          implicit none
51          integer, parameter :: i8 = selected_int_kind(8)
52          integer(i8), intent(in)  :: i1, i2
53          integer(i8), intent(out) :: i3
54  ! local variables
55          integer :: i
56          integer, dimension(8) :: bin_i1, bin_i2, bin_i3
57
58          call dec_to_bin(i1, bin_i1)
59          call dec_to_bin(i2, bin_i2)
60  ! XOR implementation
61          do i = 1, 8
62              if (bin_i1(i) /= bin_i2(i)) then
63                  bin_i3(i) = 1
64              else
65                  bin_i3(i) = 0
66              endif
67          enddo
68
69          call bin_to_dec(bin_i3, i3)
70      end subroutine XOR
71  end module xor_module
72
73  program main
74      use xor_module
75      implicit none
76
77      integer(i8) :: i1, i2, i3
78      integer, dimension(1:8) :: bin_i1, bin_i2, bin_i3
79
80      print *, "input for i1: "
81      read *, i1
82      print *, "input for i2: "
83      read *, i2
84      print *, "Your input for i1: ", i1, "i2: ", i2
85      call dec_to_bin(i1, bin_i1)
86      call dec_to_bin(i2, bin_i2)
87
88      print '(A, i4, a, 8i1)', "binary conversion of ", i1, " is: ", bin_i1(1:8)
89      print '(A, i4, a, 8i1)', "binary conversion of ", i2, " is: ", bin_i2(1:8)
90      print *, "calling subroutine XOR"
91      call XOR(i1, i2, i3)
92      print *, "i3 is ", i3
93      call dec_to_bin(i3, bin_i3)
94
95      print '(A, i4, A, 8i1)', "binary conversion of ", i3, " is: ", bin_i3(1:8)
96  end program main
```

Question 2. Write code `main02.f90`, which works as below:

- Create `point` module that contains two coordinate values of x and y.
- Create `triangle` module that contains three `point` module in it.
- Create a function called `area` in a `triangle` module and write a function that returns the area of a triangle formed by three points.

- When there are two `triangle` modules `a` and `b`, if you do `a+b`, add the area of the two triangles `a, b` and return them.
- As for the coordinates of one triangle in the code, set the three points as (1,1) (4,1) (1,5) and check that the area comes out to be 6. `triangle t1`
- As for the coordinates of one triangle in the code, set the three points as (0,0) (1,0) (0.5,0.866) and check the area comes out to be approximately 0.433. `triangle t2`
- check that the result of `(t1+t2)` is approximately 6.433.
- Leave the check process in the code.

My answer:

```
! 2024.5.26. Jeongbin Kim
! N.E. Engineering Programming midterm5 (Fall 2021)
! Problem 2, Calculation of area when 3 point is given, using user-derived types.

module point_module
    implicit none

    type point
        real :: x, y
    end type point
end module point_module


module triangle_module
    use point_module
    implicit none

    type triangle
        type(point) :: a, b, c
    end type triangle

!    interface operator (+)
!        module procedure sum_area
!    end interface operator (+)

contains
    real function area(t) result(triangle_area)
        implicit none

        type(triangle), intent(in) :: t

        triangle_area = 0.5*( t%A%x*(t%B%y-t%C%y) + t%B%x*(t%C%y-t%A%y) + t%C%x*(t%A%y-t%B%y) )
    end function area
!    real function sum_area(t1, t2)T
!        type(triangle), intent(in) :: t1, t2
end module triangle_module

program main
    use triangle_module
    implicit none

    type(triangle) :: t1, t2
    real :: area_t1, area_t2
    integer :: input

    print *, "Type ' 1 'to select automated procedure", achar(10), "Or:"
    print *, "Type ' 2 ' to select manual procedure", achar(10)

    read(*,*) input

    if (input == 1) then
        t1%A%x = 1.0
        t1%A%y = 1.0
        t1%B%x = 4.0
        t1%B%y = 1.0
        t1%C%x = 1.0
        t1%C%y = 5.0
```

```
        t2%A%x = 0
        t2%A%y = 0
        t2%B%x = 1.0
        t2%B%y = 0
        t2%C%x = 0.5
        t2%C%y = 0.866
        area_t1 = area(t1)
        area_t2 = area(t2)
        print *, "area of t1 is: ", area_t1
        print *, "area of t2 is: ", area_t2
        print *, "Addition of these 2 areas equals to: ", area_t1 + area_t2
    else if (input == 2) then
        print *, "For triangle t1: "
        print *, "For point A, input x and y"
        read *, t1%A%x, t1%A%y
        print *, "For point B, input x and y"
        read *, t1%B%x, t1%B%y
        print *, "For point C, input x and y"
        read *, t1%C%x, t1%C%y

        area_t1 = area(t1)
        print *, "area of t1 is: ", area_t1

        print *, "For triangle t2: "
        print *, "For point A, input x and y"
        read *, t2%A%x, t2%A%y
        print *, "For point B, input x and y"
        read *, t2%B%x, t2%B%y
        print *, "For point C, input x and y"
        read *, t2%C%x, t2%C%y

        area_t2 = area(t2)
        print *, "area of t2 is: ", area_t2

        print *, "adding areas of 2 triangles :", area_t1 + area_t2
    else
        print *, "Wrong input!", achar(10), "Terminating program."
    endif

end program main
```

Question 3. Write the code `main03.f90` as following:

- Construct a subroutine `PrintBinary(i1)` to console out binary representation of integer `i1` in decimal representation where `i1` is a 32-byte integer using recursion call. **Do not use loop.** The procedure is explained as below:

```
    PrintBinary(13)
        PrintBinary(6)
            PrintBinary(3)
                PrintBinary(1)
                Console out 1
            Console out 0
        Console out 1
    Console out 1
```

- Therefore, the final result have to be `1101`.
- Check that the result of (`PrintBinary(13)`) is 1011 and (`PrintBinary(213)`) is 11010101.
- Leave the check process in the code.

My answer:

```
! 2024.2.25. Jeongbin Kim
! N.E. Engineering Programming midterm5 (Fall 2021)
! Problem 3, Conversion of decimal point to binary using recursive subroutine

program main
    implicit none
```

```
7       integer, parameter :: int8 = selected_int_kind(8)
8       integer(int8) :: i1
9
10      print *, "input number "
11      read *, i1
12      call PrintBinary(i1)
13
14  contains
15      recursive subroutine PrintBinary(i1)
16          implicit none
17          integer(int8), intent(in) :: i1
18          integer :: i1_local
19
20          i1_local = i1
21          if ( mod(i1_local, 2) == 0 .and. i1_local /= 0) then
22              i1_local = i1_local / 2
23              call PrintBinary(i1_local)
24              print *, "0"
25          else if ( mod(i1_local,2) == 1 .and. i1_local /= 0) then
26              i1_local = i1_local / 2
27              call PrintBinary(i1_local)
28              print *, "1"
29          endif
30      end subroutine PrintBinary
31
32  end program main
```